Bolt Beranek and Newman Inc.

bbn

12

B.S.

AD A053959

Report No. 3806

Planning for ACCAT Remote Site Operations

Quarterly Technical Report No. 2, 15 September 1977 to 15 December 1977

April 1978

Prepared for:
Defense Advanced Research Projects Agency

BBN Report No. 3806

Planning for ACCAT Remote Site Operations

Quarterly Technical Report No. 2
15 September 1977 to 15 December 1977

Sponsored by:
Defense Advanced Research Projects Agency
ARPA Order No. 3175.8

Monitored by:
Naval Electronic Systems Command
Contract No. N00039-77-C-0239

Contract Period: 14 June 1977 to 13 June 1978

Principal Investigator: Robert H. Thomas

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

14 BBN [redacted]-3806

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

6 Planning for ACCAT Remote Site Operations

5. TYPE OF REPORT & PERIOD COVERED

9 Quarterly Technical rept. no. 2,
[redacted]

6. PERFORMING ORG. REPORT NUMBER

15 Sep 77 - 15 Dec 77

7. AUTHOR(s)

R. Thomas
P. Johnson

10 Robert/Thomas
P./Johnson

8. CONTRACT OR GRANT NUMBER(s)

15 N00039-77-C-0239,
ARPA Order-3175[redacted]

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts 02138

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

12. REPORT DATE

11 Apr[redacted]78

13. NUMBER OF PAGES

9 12 11p.

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report)

Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 3175.8.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Advanced Command Control Architectural Testbed (ACCAT)
command control
ARPANET
Remote Site Modules
TENEX Operating System
interprocess communication

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes BBN efforts to perform site surveys and planning for the installation of ACCAT remote site modules at selected sites; to provide general system architecture and design services for the ACCAT program; and to assist the ARPA staff in the planning, maintenance, and conducting of demonstrations of various ARPA computer and communication technologies.

DD FORM 1473, 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

069 100

TABLE OF CONTENTS

1. Introduction

This is the second Quarterly Technical Report for this contract. It reports on project activity for the period between September 15, 1977 and December 15, 1977.

The Advanced Command Control Architectural Testbed (ACCAT) is a facility designed to support evaluation of the applicability of various new computer-communication and information processing techniques to military command and control problems. The ACCAT program is sponsored jointly by ARPA and the Navy.

The core of the ACCAT facility is located at the Naval Ocean Systems Center (NOSC) in San Diego. It began operation in mid-1977. The testbed is built on a number of existing capabilities including: the ARPANET; the ability to provide secure communication for subnetworks within the ARPANET; the standard interfaces and protocols of the network which enable interoperability of heterogeneous equipment; and a large base of existing software and experience in computer networking, time-sharing and interactive computing.

The ACCAT concept includes support for remote site operations. Initially, this will involve secure access from distant locations to the core ACCAT facility at NOSC. At a later time, the ACCAT resources may be enhanced with the addition of computing capability at one or more of these remote sites. ACCAT

activity at a given remote site will be via a "remote site module" (RSM).

The object of this project is to perform site surveys and planning for the installation of ACCAT remote site modules at selected sites; to provide general system architecture and design services for the ACCAT program; and, to develop a plan for making (selected) services of the Fleet Numerical Weather Center (FNWC) available to the ACCAT facility through the FNWC remote site module. In addition, as part of this project, we are assisting the ARPA office in planning, maintenance, and conduct of demonstrations of various ARPA information processing technologies.

Project activity during the quarter included the following:

- Site survey activity for the Naval Postgraduate School (NPS) continued. A second visit to NPS was made to confer with NPS personnel regarding site preparation requirements and to consider alternative equipment layout plans. The NPS site survey is substantially complete, and a detailed site survey report is being prepared.

- We have started a UNIX implementation of MSG, the interprocess communication facility developed for the National Software Works (NSW) system. At a July 1977 meeting, hosted by the RAND Corporation, MSG was adopted as the standard for ACCAT

interprocess communication. Implementations of MSG already exist for TENEX and TOPS-20. When the UNIX MSG implementation is completed, all of the ACCAT host systems will support MSG. The bulk of the implementation work will be performed during the next two periods.

- We met with personnel from Massachusetts Computer Associates (MCA) to help finalize plans for installation of the ACCAT remote site module at the Fleet Numerical Weather Center (FNWC). Discussions focused on the hardware and software interface between the Print Line Interface (PLI) and the MCA PDP-11 front end for the FNWC computers.

- We assisted in a demonstration of ARPA information processing technologies conducted at SRI in September for Army personnel. This demonstration emphasized the Packet Radio Project.

- We attended the ACCAT principal investigator meeting at Woods Hole in September. At that meeting we discussed the FNWC and NPS site planning activities, the UNIX MSG implementation, and the potential role NSW might play in ACCAT.

The remainder of this report discusses the UNIX MSG implementation effort in more detail.

2. Implementation of MSG for UNIX

In designing the implementation of MSG for UNIX, we started with our designs for the TENEX and TOPS-20 implementations. We modified them as necessary to take advantage of special UNIX features and to avoid TENEX/TOPS-20 features absent from UNIX.

The implementation approach remains basically the same. There is a collection of processes to handle overall control and communication with MSG modules on other hosts. This collection is known as the "Central MSG." For each user process communicating via MSG there is a process which implements the interface between the user process and MSG. This process is known as a "Process Controlling MSG" or a "Local MSG." In addition, it functions to allow efficient intra-host MSG communication and to protect the MSG system from aberrant or malicious user programs.

The TENEX and TOPS-20 implementations rely heavily upon the flexible interprocess communication (IPC) mechanisms available on those hosts, and on the ability of processes to share address spaces. These IPC mechanisms are used to support communication among local MSGs, and between them and the central MSG. On UNIX alternate IPC mechanisms must be used. It is in this area that the standard UNIX system is inadequate. The basic UNIX IPC mechanism is the "pipe." A pipe allows two processes to exchange

data streams after some prior arrangement has been made by a common ancestor. Furthermore, there is no structure imposed on communication over a pipe. Since several processes may share a pipe, a higher level communication protocol must be used to enable a receiving process to distiguish between data sent to it over a single pipe by more than one sending process.

Recognizing these deficiencies, the RAND Corporation has developed an alternate IPC mechanism known as "Ports." Ports are an extension to the pipe mechanism. Port names are managed within the UNIX directory structure. Use of the directory structure for port naming allows two unrelated processes to communicate with each other, with protection provided by the normal UNIX file access control mechanisms. In addition, a message structure is imposed on communication over ports. This is accomplished by adding a message header to data sent over a port. The receiver may read the header to determine the identity of the sending process.

The port mechanism is adequate to support communication among the local MSG processes and between them and the central MSG process.

The other major difficulty with standard UNIX is the fact that all I/O operations are blocking. That is, if a read operation is attempted when no input is available, the process

reading will block until data is available. Similarly, a process that attempts a write operation when no capacity is available (e.g., a pipe or port is full) will block until room is made available. This makes it difficult, if not impossible, to efficiently implement a process, such as the central MSG, whose basic task is coordinating information flow between many other processes over several information channels.

The UNIX group at BBN developing a UNIX implementation of TCP has recognized this problem, and has developed two new system calls for UNIX to overcome it. The first (CAPAC) allows a process to determine how much data, if any, can be safely read or written on a given data channel. The second (AWAIT) allows a process to wait for activity on any of several data channels. Together these two operations allow a process to monitor activity over a number of communication channels with other processes and safely (i.e., without danger of being blocked by a non-cooperating or malfunctioning process) act as a central coordinator of information flow between them. This is essentially MSG's role.

The design for the UNIX is thus as follows. Associated with each process that uses MSG are a set of process interface routines that the user process can call to cause MSG operations to occur. These routines do most of the parameter checking needed and then communicate over a pipe with a parallel process

which acts as the local MSG. The local MSG is responsible for communication with the central MSG over a port, with other local MSGs over ports for intra-host MSG communications, and for protecting the MSG system from interference by user processes. A central process (or collection of processes) acts as the central MSG to coordinate MSG activity on the host. It handles all communication with MSGs on other hosts, allocates various MSG resources such as process names, creates new MSG processes as necessary to handle received Generic messages, etc.

Since UNIX allows a process to derive its identity for access control purposes from the file being run in it, it is possible for the local MSG and the central MSG to access system resources that are inaccessible to the user processes communicating via MSG. This makes it relatively easy to implement protection for internal MSG data bases and the communication ports used by MSG. This feature also makes it straightforward to concurrently run multiple MSG systems in an independent manner on the same UNIX host.

The current state of the implementation is as follows.

To help design the user process interface to MSG, two standard MSG test programs (the so-called "M1" and "M2" programs; see BBN Report 3752 for details) were written with dummy MSG interface routines. These test programs have been debugged and

will be used to test and debug interim and final implementations of MSG.

As the result of this exercise, the interface between user processes and MSG has been designed. The process/MSG interface closely parallels that of the MSG Design Specification, with messages, process names, dispositions, and so forth, being passed between MSG and the process address space. The primary divergence from the specification is with "signals." (A signal is the mechanism by which MSG notifies user processes that an operation, such as ReceiveMessage, has completed.) With the exception of block/unblock and the "flag" signal (which requires that a process poll to detect it), none of the signal examples in the Design Specification are appropriate for UNIX. A new type of MSG signal is provided for UNIX. For this the process will make a special call (RequestSignal()) when it is prepared to block and wait for an MSG event to complete. When this occurs, the RequestSignal() call will return with a "pending event ID" that identifies the operation which has completed. It may be appropriate in the future to add other signalling mechanisms, and the implementation is being done with the need for such flexibility in mind.

The process interface routines themselves have been designed and implemented. In so doing, the interface between these routines and the local MSG was designed.

The local MSG was next designed and is currently being implemented. Finally, the central MSG has been partially designed; its interface with the local MSG has been specified and its general structure has been defined.